

Type- and Token-based Word Embeddings in the Digital Humanities

Anton Ehrmanntraut, Thora Hagen, Leonard Konle and Fotis Jannidis

Julius-Maximilians-Universität Würzburg

Abstract

In the general perception of the NLP community, the new dynamic, context-sensitive, token-based embeddings from language models like BERT have replaced the older static, type-based embeddings like word2vec or fastText, due to their better performance. We can show that this is not the case for one area of applications for word embeddings: the abstract representation of the meaning of words in a corpus. This application is especially important for the Computational Humanities, for example in order to show the development of words or ideas. The main contribution of our papers are: 1) We offer a systematic comparison between dynamic and static embeddings in respect to word similarity. 2) We test the best method to convert token embeddings to type embeddings. 3) We contribute new evaluation datasets for word similarity in German. The main goal of our contribution is to make an evidence-based argument that research on static embeddings, which basically stopped after 2019, should be continued not only because it needs less computing power and smaller corpora, but also because for this specific set of applications their performance is on par with that of dynamic embeddings.

Keywords

Word Embeddings, BERT, fastText

1. Introduction


Since 2013 word embeddings and modern language models have revolutionized the representation of words, sentences and texts in natural language processing. This started with the now famous word2vec [26] algorithms in 2013 and developed in a fast progression with the models showing an astonishing increase in performance over recent years. While the first generation of word embeddings was type-based, mapping each token to a vector, blending all contexts it was used in, the next generation with models like BERT [10] is context-sensitive, mapping the same token to different vectors, depending on the context the token is present. These token-based word embeddings are now the state of the art in natural language processing. But these models have increased dramatically in size and need much more computing power which makes it difficult or even impossible to train them from scratch in a typical digital humanities setup. These pragmatic reasons were the motivation for the research we will describe: we

CHR 2021: Computational Humanities Research Conference, November 17–19, 2021, Amsterdam, The Netherlands

✉ anton.ehrmanntraut@uni-wuerzburg.de (A. Ehrmanntraut); thora.hagen@uni-wuerzburg.de (T. Hagen); leonard.konle@uni-wuerzburg.de (L. Konle); fotis.jannidis@uni-wuerzburg.de (F. Jannidis)

🆔 0000-0001-6677-586X (A. Ehrmanntraut); 0000-0002-3731-6397 (T. Hagen); 0000-0001-5833-0414 (L. Konle); 0000-0001-6944-6113 (F. Jannidis)

© 2021 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

CRedit Roles: Anton Ehrmanntraut: Investigation, Software, Writing – original draft; Thora Hagen: Data Curation, Writing – original draft; Leonard Konle: Data Curation, Writing – original draft; Fotis Jannidis: Conceptualization, Supervision, Writing – review & editing

wanted to be able to describe the loss of information and performance that results from using static instead of dynamic embeddings. But our surprising preliminary results were confirmed by a paper which was published as preprint during our work [23]: Static word embeddings can be on par with dynamic embeddings or even surpass them in some constellations. The usage of word embeddings in DH can be categorized in two groups: Using embeddings as an improved form of *word representation* in tasks like sentiment analysis [43], word sense disambiguation [29, 38], authorship attribution [19, 33] etc. And using word embeddings as *abstractions of semantic systems* by describing word meaning as a set of relation of a focus term to its semantic neighbours. Since Kulkarni et al. [21] proposed the comparison of embeddings trained on texts from different slices in time to measure semantic change, their method known as diachronic, temporal or dynamic word embeddings [22] has been adopted by the Digital Humanities community [31, 32, 16]. However, word embeddings as abstractions of semantic systems do not only work in the historical dimension, but are universally applicable. Even a comparison across several languages is possible [34].

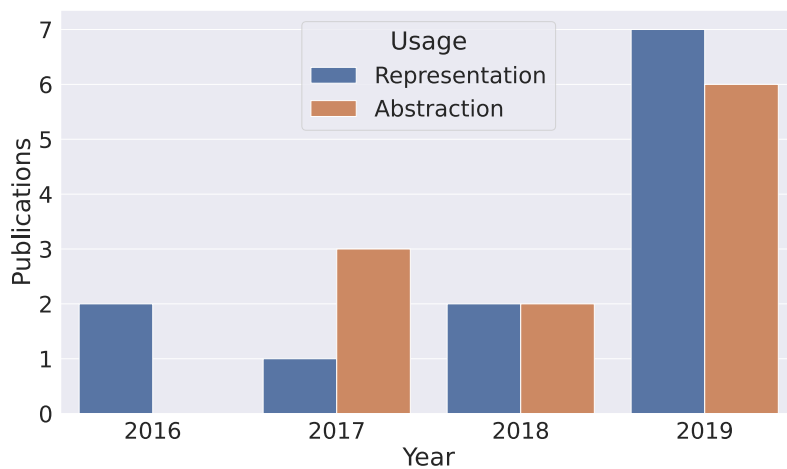


Figure 1: Publications using word embeddings as either representation or abstraction in *Digital Scholarship in the Humanities*, *DH Quarterly*, the *Digital Humanities Conference Abstracts* and its German branch *Digital Humanities im deutschsprachigen Raum*.

While token-based embeddings succeed in representation tasks, this is less clear for abstraction, since these capabilities are not included in common benchmark tests [40] and some comparisons are marred by models being trained on corpora of different sizes and the usage of different dimensions for the embeddings. At the same time, unlike in computer science, abstraction is not a rare application in DH research (see Figure 1). This poses the question under which circumstances it is worthwhile for a digital humanist, who is interested in using word embeddings as an abstraction, to work with these latest token-based approaches, if there is performance loss if one is using a token-based model and how large the loss is. To answer these questions we will create type-based embeddings from a pre-trained BERT (GBERT [8]) and compare them to static type-based embeddings, which we created. Therefore we will report on two sets of experiments: First, we will try to find a strong way to create performant type-based embedding out of token-based embeddings. Secondly, we will compare this derived embedding against static, traditional type-based embeddings. As far as possible we will train these embeddings on the same or similar corpora and compare embeddings with the same dimensions to level the playing field and avoid the distortions which have limited the usefulness of some

other comparisons. Because word embeddings as abstractions of semantic systems have a close link to questions of word similarity and word relatedness we will limit our evaluation to these aspects. In contrast to most of the existing research comparing different word embeddings, we will use German corpora for the training of the models and a German BERT model (GBERT) pre-trained on similar corpora. This makes it necessary to create our own evaluation datasets, some of them mirror English datasets, sometimes by translation, to allow an easy comparison of results across languages, some are new reflecting our interests in specific aspects of word similarity and word relatedness.

2. Related work

2.1. Creating types

Since the initial presentation of BERT, there were efforts to convert BERT’s contextualized token-based embedding (that is, the output of the respective Transformer layers from a pre-trained BERT model under certain input sequences) into conventional static, decontextualized type-based embeddings. We follow the terminology used in the survey by Rogers et al. [30] and exclusively use the term *distillation* to refer to this procedure¹.

Almost all approaches aggregate the token embeddings across multiple contexts into a single type embedding in some way [10, 6, 39, 23]. Bommasani et al. [6] present a natural and generalized description of the distillation process, which covers all previously cited approaches. In Section 4, we present and extend their description, and examine a wider range of possible parameters. In contrast to the experiments of Bommasani et al., we also included an evaluation on the basis of relations resp. analogies. Also, different to the experiments of Vulić et al. [39] and Lenci et al. [23], we experiment with aggregations beyond the component-wise arithmetic mean.

Similar to previous works, we only compute the embedding for a small selection of types relevant for our evaluation datasets. Thus, we already remark at this point that the generated type-based embedding is not “full” – in the sense that we assign only vectors to types that are present in our evaluation set, not to all types in the vocabulary, as is the case in static type-based embeddings. This has consequences on the type of tasks we can use to probe this small embedding, hence we were required to reformulate some tasks to account for this limit.

The techniques independently proposed by Wang et al. [42], and Gutman and Jaggi [13], resp., appear to be the only processes that are not a special case of the generalized description by Bommasani et al. In both works, contextualized embeddings from BERT are used to complement the training of a word2vec-style static embedding. Wang et al. use BERT to replace the center word embeddings in a skip-gram architecture, while Gutman and Jaggi use BERT to replace the context embeddings in a CBOW architecture. However, these techniques come with large computational effort, since training the static embedding requires at least one full BERT embedding of the entire training corpus. Therefore, we omit a detailed analysis of this technique, since this degree of required computational resources seems out of reach for a DH setup.

¹Unfortunately, this terminology might be misleading. In particular, it should not be confused with the compression technique called “knowledge distillation” utilized in, e.g., DistilBERT.

2.2. Evaluation of type-based word embeddings

Word embeddings are evaluated either intrinsic with their ability to solve the training objective or extrinsic by measuring their performance on other NLP problems [4]. Since the development of BERT, problems using word embeddings as representations of words or sequences rather than abstractions (see Section 1) to perform supervised training on curated datasets are dominant in evaluation. Those benchmarks like GLUE [41, 42] are not in scope of this paper, because we are solely interested in abstraction. From the abstraction viewpoint word embeddings should represent various linguistic relationships between the words [41]. These relationships are distributed over several datasets to test vector spaces for desired properties. These datasets typically contain tests on: Word Similarity, Relatedness, Analogies, Synonyms, Thematic Categorization, Concept Categorization and Outlier Detection [4, 41].

For word relatedness, pairs of words are given a score based on the perceived degree of their connection, which are then compared to the corresponding distances in the vector space. For the word similarity task specifically, the concept of relatedness is dependent on the degree of synonymy [36]. Some of the most prevalent word relatedness/similarity datasets for the English language, among others, are WordSim-353 [2], MEN [7] and SimLex-999 [18]. While WordSim-353 and MEN focus on relatedness, SimLex-999 has been specifically designed to represent similarity, meaning that pairs rated with high association in MEN or WordSim-353 could have a low similarity score in SimLex-999, as “association and similarity are neither mutually exclusive nor independent” [18]. Additionally, SimLex-999 includes verbs and adjectives apart from nouns, which the other two datasets do not. Another constraint of the MEN dataset is that there are no abstract concepts present in the word pairs.

Both relatedness and similarity can also be evaluated via the word choice task, where each test instance consists of one focus word and multiple related or similar words in varying (relative) degrees. Concerning word choice datasets, the synonym questions in the TOEFL exam are the most prominent. One test instance consists of a cue word and four additional words, where exactly one is a true synonym of the cue word. The distractors are usually related words or words that could generally replace the cue word in a given context, but would change the meaning of a sentence [35]. As these questions have been constructed by linguists, the data reliably depicts word similarity. However the design of the dataset does not allow for distinguishing medium or low similarity for example because of the binary classification approach as opposed to the WS task [18].

Lastly, the word analogy task can be used to probe specific relations in the vector space. Given two related terms and a cue term, a target term has to be predicted analogous to the relation of the given word pair. For word analogy datasets, the Google Analogies test set [26] is the most notable. It includes five semantic and nine syntactic relations, where the semantic relations mostly cover world knowledge (e.g. countries and capitals), while the morphological relations include the plural of nouns and verbs or comparative and superlative among others.

The usual implementation to test embeddings on this task uses linear vector arithmetic. For example, given analogy “*man* is to *woman* as *king* (the cue term) is to *queen* (the target term)”, we test if *queen* is the closest type in the vocabulary to $king - man + woman$. This implementation builds upon the supposition that the underlying embedding exhibits linear regularities among these analogies – in above example, that would be $woman - man \approx queen - king$, i.e. there is a prototypical “womanness”-offset in the embedding. [27, 25]

Since it was computationally infeasible for us to distill embeddings from BERT that have a comparable vocabulary size to those of static embeddings, we found that this setup becomes

unreliable: Due to the smaller vocabulary, we heavily restrict the search space in these analogy tests, making the prompts appear easier to solve. Following the example, consider vector $\mathbf{v} = \textit{king} - \textit{man} + \textit{woman}$. Due to the small vocabulary, there are only few distractors in the neighborhood of \mathbf{v} . Consequently, the vector *queen* most probably is the closest type from the vocabulary and the prompt is answered correctly, but this is not a cause of a structurization of the embedding space.

3. Resources

3.1. General Corpora

We train the type-based embeddings on the German OSCAR Corpus (Open Super-large Crawled Aggregated coRpus [28]), as this is the largest chunk of the training data for the current best German BERT model discussed below. The deduplicated variant of the German corpus contains 21B words (145 GB), filtered out of CommonCrawl.²

3.2. BERT Model

As outlined in the introduction, we want to ensure that the different language models are trained on the same or similar corpora. While training of type-based models on our chosen corpora were feasible for us, it was impossible to pre-train a BERT model from scratch. Therefore, we choose a pre-trained German model GBERT_{Base} provided by Deepset in collaboration with DBMDZ [8]. Like the original BERT_{Base}, the German model consists of 12 layers, 768 dimensions, and a maximum sequence length of 512 tokens. Also, we choose this model since, to date, this trained model appears to be the currently best available BERT model (with above hyperparameters) [8]. The model was trained on a combination of four corpora, whereas OSCAR dominates the training set by approximately 88% of the total data. When we distill type-based embeddings from BERT, we always are going to use the GBERT_{Base} model, and will only use the OSCAR corpus to retrieve contextualized inputs. Likewise, when we train static type-based models, we are only going to use the OSCAR corpus (neglecting the remaining 22% of BERT’s pre-training data unaccounted for).

3.3. Evaluation Data

As the most popular evaluation datasets are in English, we constructed a comprehensive, German test suite consisting of multiple datasets which cover different aspects based on already existing evaluation data.³ The tasks covered are: word relatedness (WR), word similarity (WS), word choice (WC) and relation classification (RC). In addition, the data probes semantic knowledge such as synonyms and morphological knowledge, namely inflections and derivations. See Table 1 for an overview of all test datasets.

Word Relatedness/Similarity For WR, we used the re-evaluated translation of WordSim-353 (Schm280) as presented in [20], where we only corrected nouns which were written in lower case, as well as a DeepL translation of [7] (MEN), which we then reviewed and adjusted manually as needed. To assess WS, we opted for the translation of [18] (SimLex999) by [24]. Both

²<https://commoncrawl.org>

³Datasets and Code: <https://github.com/cophi-wue/Word-Embeddings-in-the-Digital-Humanities>

WR and WS are judged via the Spearman’s rank correlation coefficient between the human annotated scores and the cosine distance of all word pairs.

Relation Classification As outlined above, we incorporate the concept of linearly structured word analogies into the test suite not in the usual way, but using relation classification. Instead of predicting a target word through a given cue word, RC tries to predict the relation type of a word pair by comparing their offset with representative offsets for each relation type. Specifically, we are given a collection of relations R_1, R_2, \dots, R_k where each R_i is a set of word pairs. We now interpret the relation R_i as a set of offsets: for each word pair $(a, b) \in R_i$, we consider the vector $\mathbf{v}_b - \mathbf{v}_a$. For the evaluation, we use a median-based 1-nearest-neighbor classification: We “train” the classifier by choosing the median of R_i as decision object. More precisely, we define vector $\mathbf{r}_i = \text{median}(\{\mathbf{v}_b - \mathbf{v}_a \mid (a, b) \in R_i\})$ as decision object of R_i . We then test this classifier on all pairs from all relations, thus check for each pair (a, b) from R_i whether \mathbf{r}_i is in fact the closest decision object to $\mathbf{v}_b - \mathbf{v}_a$ (with respect to ℓ_1 -norm). We evaluate these predictions by the “macro” F1 score, i.e. the unweighted average of the F1 scores under each relation type, respectively.

While this setup allows for different aggregations resp. distance functions other than median resp. ℓ^1 -norm, we surprisingly found this choice to be more successful than other candidates (such as those based on cosine distance) among all examined embeddings.

For the RC data we made use of two German knowledge bases: the knowledge graph GermaNet [15, 17] (Ver. 14.0) and the German Wiktionary.⁴ GermaNet incorporates different kinds of semantic relations, including lexical relations such as synonymy, and conceptual relations such as hypernymy and different kinds of compound relations. For the RC evaluation, we only selected the conceptual relations and the pertainym relation for our GermaNet dataset, since only these can be considered a directed one-one relation. Wiktionary on the other hand contains tenses, the comparison of adjectives, and derivational relations among other morphological relations. Again, we selected a set of inflectional resp. derivational directed one-one relations for the Wiktionary dataset for the RC evaluation, cf. Table 7 in the appendix. Even though there is a German version of the Google Analogies dataset available [20], we chose not to include it, as its semantic and morphological relations are covered entirely by GermaNet and Wiktionary, respectively. Additionally, both datasets contain more instances than the Google Analogies testset does.

Word Choice Lastly, we included the WC task. Here, we used a translated version of the TOEFL synonym questions [20] as well as one automatically constructed dataset from the German Duden of synonyms. Our Duden dataset includes one synonym of the cue word as the target word, plus, as distractors, four synonyms of the target word that are not synonyms of the cue word. Evaluation is based on whether the target word is closer to the cue word than the distractors with respect to cosine distance. We report accuracy among all prompts.

Initially, we also wanted to explore world knowledge (i.e. named entities) captured by the embeddings, such as city-body of water or author-work relationships. However most named entities consist of multi-word expressions which are difficult to model via type or token based embeddings. We therefore removed all instances where a concept consisted of more than one token in the datasets described above.

Table 1

Datasets and corresponding tasks in the embedding testsuite

| dataset | GermaNet | Wiktionary | SimLex999 | Schm280 | MEN | Duden | TOEFL |
|-----------|-------------------|--------------------|-----------|---------|------|-------|-------|
| task | RC | RC | WS | WR | WR | WC | WC |
| instances | 3980 (42 rel.) | 23174 (33 rel.) | 996 | 279 | 2964 | 1184 | 401 |

Table 2

Examined Vectorization, Subword pooling, and Context Combination functions

| Vectorization | |
|--------------------------------|---|
| inputemb | use BERT’s pretrained input embedding as subword vector |
| L1, ..., L12 | output of layer N |
| all | concatenation of the vectors L1, ..., L12 |
| L1-4 / L9-12 | concatenation of the vectors L1, ..., L4 / L9, ..., L12 |
| Subword Pooling | |
| first | take first subword vector as token vector |
| mean | component-wise arithmetic mean |
| median | component-wise median |
| meannorm | mean-norm of all subword vectors as token vector (see Eq. 1) |
| l0.5medoid, l1medoid, l2medoid | medoid of the subword vectors w.r.t. $\ell^{0.5}$ -distance, ℓ^1 -distance, ℓ^2 -distance |
| nopooling | skip the subword pooling stage and consider each subword vector as individual token vector |
| Context Combination | |
| mean | component-wise arithmetic mean as type vector |
| median | component-wise median as type vector |
| meannorm | mean-norm of all token vectors as type vector (see text) |
| l0.5medoid, l1medoid, l2medoid | medoid of the subword vectors w.r.t. $\ell^{0.5}$ -distance, ℓ^1 -distance, ℓ^2 -distance |

4. Creating Type Vectors

Comparing embeddings distilled from BERT’s token-based embeddings with traditional static type-based embeddings requires us to examine different possibilities on how to perform this distillation. Therefore, we compare these possibilities by evaluating the resulting embeddings on the discussed evaluation dataset. Given these results, we decide on a single distillation procedure and compute a BERT embedding to compare its performance against static embeddings in Section 5.

In order to systematically evaluate the different methods to compute an embedding from BERT’s token-based representations, we follow and extend the two-stage setup of Bommasani et al. [6]. The first stage is the subword pooling, where the k subword vectors $\mathbf{w}_c^1, \dots, \mathbf{w}_c^k$ (derived from BERT’s output) for word w in context c are aggregated into a single contextualized token vector with aggregation function f ; that is, $\mathbf{w}_c = f(\{\mathbf{w}_c^1, \dots, \mathbf{w}_c^k\})$ of w in c . Then, in the second stage, the context combination, multiple contextualized token vectors $\mathbf{w}_{c_1}, \dots, \mathbf{w}_{c_n}$ are aggregated by function g into a single static embedding $\mathbf{w} = g(\{\mathbf{w}_{c_1}, \dots, \mathbf{w}_{c_n}\})$.

We extend this description by prepending a zeroth vectorization stage, in which we make explicit how to transform the outputs of BERT’s layers into a single subword vector. While

⁴<https://dumps.wikimedia.org/dewiktionary/20210701/>

Bommasani et al. tacitly concatenate all outputs to form a subword vector, we also allow to selectively pick specific layer output(s) as subword vector, or a summation of the layers.

This setup also allows us to choose pooling functions f resp. context combination functions g which are not defined component-wise. Hence, we examine a wider range of choices for vectorizations, f and g than previously considered, e.g., in [39, 6]. One such considered novel aggregation function is *mean-norm*, which refers to the aggregation

$$\text{meannorm}(\mathbf{v}_1, \dots, \mathbf{v}_n) = \text{norm} \left(\frac{1}{n} \sum_{i=1}^n \text{norm}(\mathbf{v}_i) \right), \quad (1)$$

that takes a set of vectors as input, normalizes each to unit length, calculates the mean on these normalized vectors, and normalizes this mean again. We motivate this aggregation function from the fact that $\text{meannorm}(\mathbf{v}_1, \dots, \mathbf{v}_n)$ is the unique vector on the unit hypersphere that maximizes the sum of cosine similarities with respect to each $\mathbf{v}_1, \dots, \mathbf{v}_n$. Thus, mean-norm could also be understood as a “cosine centroid”. In particular, mean-norm, medoids, and aggregations based on fractional distances were included in our experiments searching for suitable distillations. Table 2 shows the functions we examined. In total, we examine 17 possible vectorizations, 8 subword pooling functions and 6 context combination functions.

4.1. Method

Intending to find best-performing distillations based on the above outlined general setup, we evaluate different choices for the “free parameters” of the distillation process as shown in Table 2.

For each word w to be embedded, we retrieve $n = 100$ sentences from the OSCAR corpus as context for w , where w occurs in a sentence of maximum sequence length of 510. If w has $< n$ but at least one occurrence, we sampled all these occurrences. Types w that did not occur in the OSCAR corpus were removed from the evaluation dataset. For each occurrence in sentence s , we construct the input sequence by adding the [CLS] and [SEP] token. The respective outputs of BERT on all layers form the input for the vectorization stage. This method of generating input sequences by sampling *sentences* largely agrees with the methods proposed by [6, 39, 23], and only differs in the sampling of sentences.

Due to the large number of possible distillations, it was computationally infeasible to construct embeddings under all distillations. Therefore, in a first experiment, we examined the quality of all considered distillations on a smaller evaluation dataset, which consists of three subsets of the MEN, the Wiktionary, the GermaNet, and the TOEFL dataset. Then, after restricting the set of potential distillations to promising ones, we perform the same experiment but with the full evaluation dataset on all five datasets. In both cases, the evaluation is performed as outlined in Section 3.3.

Also, since the scores in the respective tasks are reported in different metrics, we opt for standardizing the respective scores in each task when comparing some embeddings’ performances. Therefore, for one specific task, we consider the *standardized score* as the number of standard deviations away from the mean over all model’s scores in that task. We then can report the *mean standardized score* of some model taken over all considered tasks.

Table 3

Comparison of the embeddings computed by the medoid-based distillation methods on the full evaluation dataset. Reported numbers are the mean standardized score over all seven tasks. The score of the embedding considered in sec. 5 is the maximum and is highlighted bold.

| vectorization | | inputemb | L1 | L2 | L3 | L4 | L5 | L6 | L7 | L8 | L9 | L10 | L11 | L12 | L1-4 | L9-12 | sum | all | |
|---------------|--------------|----------|--------|--------|--------|-------|-------|-------|--------|--------|--------|--------|--------|--------|--------|-------|--------------|-------|-------|
| poolings | aggregations | | | | | | | | | | | | | | | | | | |
| | nopooling | mean | -1.372 | -1.089 | -0.482 | 0.477 | 0.852 | 0.414 | 0.277 | 0.002 | -0.182 | -0.343 | -0.137 | -0.100 | -0.618 | 0.475 | 0.096 | 1.134 | 1.122 |
| | mean | meannorm | -2.068 | -1.392 | -0.782 | 0.413 | 0.800 | 0.394 | 0.308 | 0.014 | -0.123 | -0.304 | -0.077 | -0.015 | -0.605 | 0.278 | 0.136 | 1.122 | 1.117 |
| | median | -1.631 | -1.084 | -0.679 | 0.334 | 0.795 | 0.428 | 0.228 | -0.060 | -0.206 | -0.399 | -0.064 | -0.135 | -0.582 | 0.494 | 0.171 | 1.157 | 1.091 | |
| mean | mean | meannorm | -1.372 | -1.089 | -0.482 | 0.477 | 0.852 | 0.414 | 0.277 | 0.002 | -0.182 | -0.343 | -0.137 | -0.100 | -0.618 | 0.475 | 0.096 | 1.134 | 1.122 |
| | mean | meannorm | -1.712 | -1.253 | -0.685 | 0.432 | 0.838 | 0.432 | 0.338 | 0.034 | -0.102 | -0.285 | -0.101 | -0.025 | -0.607 | 0.361 | 0.126 | 1.141 | 1.129 |
| | median | -1.301 | -1.018 | -0.516 | 0.460 | 0.854 | 0.408 | 0.259 | -0.025 | -0.210 | -0.354 | -0.075 | -0.044 | -0.514 | 0.504 | 0.133 | 1.132 | 1.132 | |
| meannorm | mean | meannorm | -2.020 | -1.373 | -0.783 | 0.370 | 0.788 | 0.384 | 0.282 | 0.013 | -0.158 | -0.293 | -0.062 | -0.053 | -0.625 | 0.259 | 0.121 | 1.109 | 1.093 |
| | mean | meannorm | -2.071 | -1.391 | -0.782 | 0.409 | 0.805 | 0.394 | 0.308 | 0.011 | -0.126 | -0.293 | -0.067 | -0.021 | -0.604 | 0.278 | 0.131 | 1.124 | 1.119 |
| | median | -1.931 | -1.272 | -0.764 | 0.348 | 0.779 | 0.350 | 0.270 | 0.001 | -0.206 | -0.351 | -0.060 | -0.057 | -0.482 | 0.307 | 0.139 | 1.080 | 1.054 | |
| median | mean | meannorm | -1.490 | -1.114 | -0.550 | 0.480 | 0.870 | 0.523 | 0.261 | -0.018 | -0.180 | -0.256 | -0.155 | -0.104 | -0.655 | 0.485 | 0.153 | 1.201 | 1.086 |
| | mean | meannorm | -1.718 | -1.270 | -0.703 | 0.437 | 0.891 | 0.554 | 0.314 | 0.046 | -0.128 | -0.235 | -0.104 | -0.024 | -0.684 | 0.390 | 0.170 | 1.170 | 1.060 |
| | median | -1.469 | -1.082 | -0.580 | 0.452 | 0.934 | 0.497 | 0.269 | -0.054 | -0.187 | -0.341 | -0.118 | -0.069 | -0.534 | 0.550 | 0.175 | 1.253 | 1.108 | |

4.2. Results

The first run of the experiment with all examined distillations on the smaller datasets gave strong indication that centroid-based distillations lead to significantly better-performing embeddings than those distillations that consist of a medoid-based poolings resp. aggregations. In fact, among the 13 distillations with the highest mean standardized score, all twelve distillations with centroid-based poolings and aggregations are present (Nopooling, mean, median, mean-norm). Numerical values are presented in Table 8 in the appendix. (Top 13 distillations are underlined.) With this experiment, we contributed insight on the performances on different aggregation functions not previously considered in the literature. The results suggest the interpretation that centroids – which represent a vector cluster by some synthetic aggregate – generally lead to better results than medoids – which represent a cluster by some member of that cluster. Also, in our distillation setup, fractional norms do not appear to give an advantage, as opposed to research that indicate that fractional distance metrics could lead to better clustering results in high-dimensional space, e.g. [1]. Hence in total, we negatively answer potential hypotheses that certain overlooked aggregation functions could lead to immediate improvements of resulting type-based embedding distilled from BERT.

Therefore, we continue our evaluation of these twelve centroid-based parameter choices in the next experiment on the full dataset. We observe that, under the restriction on centroid-based poolings and aggregations, the choice of vectorization (i.e. layer) has a much higher influence on the embedding’s performance than the actual choice of functions f and g . This supports the general hypothesis that different layers capture different aspects of linguistic knowledge [30]. Additionally, these findings demonstrate that the default suggestions $f = \text{mean}$ and $g = \text{mean}$ in the literature [6, 39, 23] generally are a reasonable choice to perform a distillation. A visualization of the embeddings’ scores on each of the full seven datasets is presented in Figure 5 in the appendix.

Again, we ranked the $17 \times 4 \times 3$ (# vectorizations \times # restricted poolings \times # restricted vectorizations) analyzed embeddings with respect to their mean standardized score over all five tasks; cf. Table 3. This leads to our observation that those embeddings based on a *sum*-vectorization outperform any of the other embeddings; hence we suspect that a vertical summation resp. averaging over all layers can provide a robust vector representation for BERT’s tokens, capturing the summative linguistic knowledge of all layers in a reasonable fash-

ion. Also, we suspect that the smaller dimensionality of the *sum*-vectorization might give the embeddings an advantage in comparison to the vectorization that concatenates all layers: the former has 768 dimensions, while the latter concatenates 12 Transformer outputs plus BERT’s input embedding, leading to $768 \times 13 = 9968$ dimensions.

To fix a single embedding for further comparison with static embeddings, we choose the embedding with the highest mean standardized score, which is the embedding based on the distillation with vectorization *sum*, the pooling $f = \text{median}$, and the aggregation $g = \text{median}$. The respective mean standardized score is highlighted bold in Table 3. Thus, when we now speak of BERT’s distilled embedding, then we explicitly mean this distillation (sum vectorization, median pooling and aggregation). Note that this embedding consists of 768 dimensions. Nevertheless, we explicitly remark that the small differences in performances do not admit the claim that the chosen distillation is a universal method that would always perform best in any scenario. Also, we want to highlight that the previously untreated median as aggregation function appears to cause some improvement, especially in the pooling stage. Due to its robustness against outliers, we recommend to always examine this aggregation in distillations of any form that convert from token-based to type-based embeddings.

5. Comparing type- and token-based embeddings

5.1. Methods

Before training the type-based embeddings models, we preprocessed the German OSCAR Corpus to ensure that models are trained on the same version of the data. Preprocessing has been done using the word tokenizer and sentence tokenizer of NLTK.⁵ Additionally, all punctuation has been removed. We trained all models using the respective default parameters and used the skip-gram model for all embeddings. We only adapted the number of dimensions and window size to create additional embedding models for comparison. While the recommended window size is 5, a window size of 2 has proven to be more effective in capturing semantic similarity in embeddings [23]. To make up for the larger dimensionality of BERT’s distilled embedding (768), we also trained static models with 768 dimensions besides the more commonly used 300 dimensions for type-based models (under the assumption that more dimensions imply a higher quality vector space). Additionally, we concatenated the embeddings in various combinations, as these “stacked” vectors often lead to better results, as presented in [3]. We experimented with all three possible tuples consisting of BERT, word2vec, and fastText (using the 768 dimension versions only) as well as one embedding where we concatenated all three. We lastly included one fastText model with 2×768 dimensions to enable a direct comparison to the stacked embeddings.

As explained above, we evaluate how well the models represent different term relations with four tasks: word similarity and relatedness, word choice, and relation classification.

5.2. Results

The first general observation looking at Figure 2 is that BERT’s distilled embedding (again, sum vectorization, median pooling and aggregation) does not perform significantly better, contrary to our expectations. In fact, the type-based embeddings seem to be capturing term relatedness and similarity even better than the token-based embeddings distilled from BERT

⁵Natural Language Toolkit (<https://www.nltk.org/>)

Table 4: Performances of the analyzed embeddings in the seven discussed tasks. Top value per task is highlighted bold.

| | GermaNet (RC) macro F1 | Wiktionary (RC) macro F1 | SimLex999 (WS) Spearman ρ | Schm280 (WR) Spearman ρ | MEN (WR) Spearman ρ | Duden (WC) accuracy | TOEFL (WC) accuracy |
|--------------------------|---------------------------|-----------------------------|-----------------------------------|---------------------------------|-----------------------------|------------------------|------------------------|
| Word2Vec Dim300 | .657 | .739 | .429 | .748 | .719 | .605 | .791 |
| Word2Vec Dim768 | .793 | .803 | .463 | .751 | .724 | .625 | .803 |
| FastText Dim300WS5 | .643 | .795 | .439 | .757 | .734 | .617 | .796 |
| FastText Dim300WS2 | .668 | .811 | .455 | .768 | .734 | .619 | .798 |
| FastText Dim768WS2 | .774 | .863 | .491 | .771 | .736 | .634 | .813 |
| BERT (sum-median-median) | .710 | .906 | .476 | .754 | .650 | .543 | .589 |

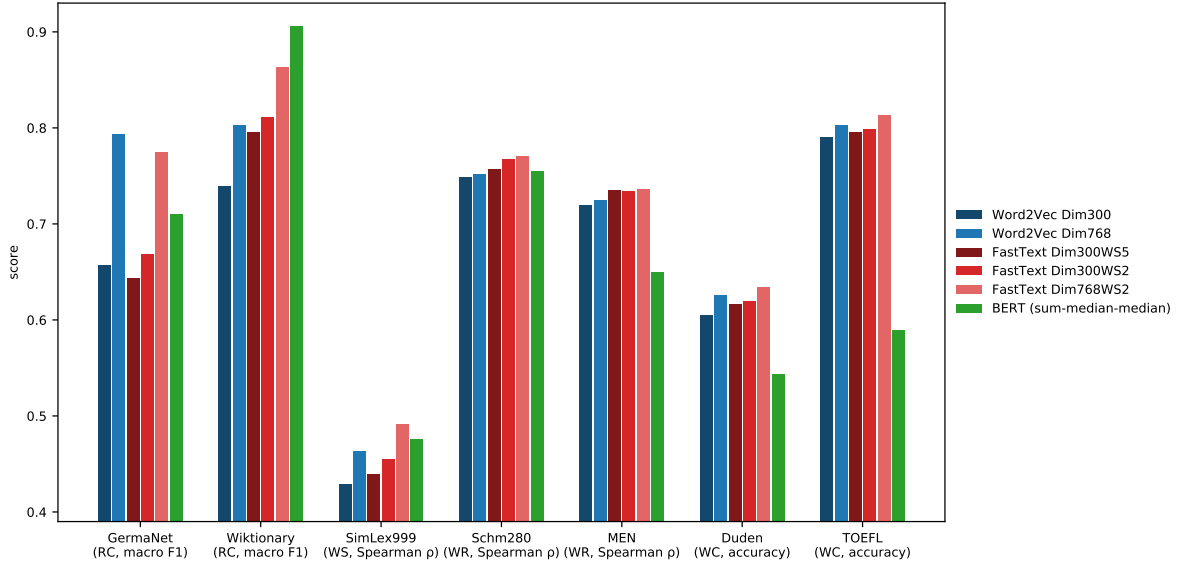


Figure 2: Performances of the analyzed embeddings in the seven discussed tasks. Graphical representation of the results in Table 4.

in most tasks: in WS, WR, and WC, FastText Dim768WS2 produces the best results (see Table 4) while in RC, BERT achieves the best results on the morphological relations only.

The WR and WS tasks (MEN, Schm280, SimLex999) paint a similar picture. Both hyperparameters, window size, and number of dimensions lead to a slight improvement when reduced and increased, respectively. Most notably, the similarity task benefits the most (about 0.05 absolute correlation improvement with both parameters adjusted for fastText, see Table 4) from altering these parameters. While BERT is on par with or even slightly outperformed by the 300-dimensional type-based embeddings in the relatedness task, it performs better in the similarity task. The higher dimensional vectors however can compare to BERT’s performance on the SimLex999 dataset. Overall, every model seems to struggle with the more narrowly defined WS task when compared to the WR task.

The WC task (Duden, TOEFL) also shows a clear trend: all type-based embeddings exceed BERT’s performance noticeably, by a 0.06 accuracy difference minimum (Duden, Word2Vec Dim300) and 0.23 maximum (TOEFL, FastText Dim768WS2). Altering the parameters of the type-based embeddings, similarly to the WR task, results in marginally better performing vectors.

BERT’s embeddings perform considerably better in the RC task when compared to the 300-dimensional embeddings. However, a substantial gain from the dimensionality increase can

also be observed with **GermaNet** as opposed to the other datasets, leading to both **FfastText Dim768WS2** and **Word2Vec Dim768** surpassing BERT’s performance by 0.06 and 0.08, correspondingly. While the same trend appears on the **Wiktionary** dataset, the classification of morphological relations by BERT’s embeddings still remains uncontested with an accuracy of 0.91. From a human perspective, the morphological relations are rather trivial (some examples are presented in Table 7 in the appendix); even from a computational point of view, lemmatizing or stemming the tails of these triples could in theory reliably predict the individual heads. This implies that generally, BERT can reproduce these kinds of simpler relations the best, while traditional models capture complex semantic associations more accurately. We separately explored the individual performances of all relations in **GermaNet** and **Wiktionary** and discovered that the higher F1 score of BERT mainly stems from the derivations, indicating that the word piece tokenization of BERT might facilitate its remarkable performance. Controlling for the dataset and relation size in a linear regression did not reveal a correlation between the amount of overlap and F1 however.

From these experiments we can conclude that for word similarity and term relatedness use cases, employing regular fastText embeddings, optionally increasing the number of dimensions, is sufficient. Using embeddings with the same number of dimensions as BERT results in the static embeddings taking the lead in the WS and RC task for semantic relations, specifically.

More so, there appears to be no clear trend on whether BERT’s distilled embedding is generally better (or worse) than others models. In certain tasks, it performs particularly well (e.g., **Wiktionary**), and in others particularly bad (e.g., **TOEFL**). To give some statistical estimate on the difference in performance between BERT and other models, we employ Bayesian hierarchical correlated t-tests proposed by Benavoli et al. [5] and Corani et al. [9], designed to compare performances of two classifiers on multiple test sets.⁶ This hierarchical model is learned on our observed scores, and after learning, can be queried to make inference on the performance difference (in score points, e.g., absolute accuracy difference) between BERT and an other language model on a *future unseen dataset*. See the cited references for a thorough presentation of the hierarchical model and the inference method. (Note that the Bayesian hierarchical correlated t-test is based on repeated cross-validation runs on the same dataset. Hence, to adapt our setup to the t-test, we need to modify our task procedures to obtain cross-validation results. Section A.2 in the appendix gives details on how we implemented this.)

Table 5 gives the results on this inference. Most prominently, it estimates that on a future unseen dataset, **FastText Dim768WS2** most certainly will outperform BERT’s distilled embedding by at least 0.03 absolute score points ($P = 89.1\%$). Even on the relatively weak **Word2Vec Dim300**, the hierarchical model predicts roughly equal probabilities for either BERT being better vs. **Word2Vec Dim300** being better (by at least 0.03 absolute score points, 47.9% vs. 51.8%).

Nevertheless, this quantitative analysis also has limits due to the stochastic model presumed by the Bayesian hierarchical correlated t-test. The model assumes that the performance differences among the datasets ($\delta_1, \delta_2, \dots, \delta_{\text{next}}$) are i.i.d. and follow the same high-level Student-t-distribution $t(\mu_0, \sigma_0, \nu)$; thus, the model assumes that the considered datasets are in some way homogeneous. Though all our datasets are meant to examine word similarity, the distinct differences in performance of the embedding types we observe (see fig. 2), indicate that these datasets represent different aspects of word similarity, which certain language models capture

⁶We want to thank the anonymous reviewer who brought the potential of the Bayesian hierarchical correlated t-test to our attention.

Table 5

Results of the Bayesian hierarchical correlated t-tests on the performance differences on a future unseen dataset. The hypotheses “BERT better” resp. “BERT worse” refer to a performance gain/loss of at least 0.03 score points. The hypothesis “practically equivalent” signifies that the performance difference between the two compared embeddings is no more than 0.03 score points.

| BERT (sum-median-median) vs. ... | BERT better | practically equivalent | BERT worse |
|----------------------------------|-------------|------------------------|------------|
| FastText Dim768WS2 | 10.05 % | 0.88 % | 89.08 % |
| FastText Dim300WS2 | 25.20 % | 0.83 % | 73.98 % |
| FastText Dim300WS5 | 34.65 % | 0.62 % | 64.72 % |
| Word2Vec Dim768 | 34.58 % | 0.33 % | 65.10 % |
| Word2Vec Dim300 | 47.95 % | 0.25 % | 51.80 % |

better than others. Hence in our use case, we see the limits of the assumptions made by the stochastic model, and in this light, the results of the Bayesian hierarchical correlated t-tests need to be interpreted cautiously.

5.3. Discussion

The most important result from our experiments is that a widespread assumption in NLP and Computational Humanities is not true: a context-sensitive embedding like BERT is not automatically better for all purposes. Static embeddings like fastText are at least on par if not better if word embeddings are used as abstractions of semantic systems. But our results are subject to some important limitations. For example, we can think of several ways which could increase BERT’s capabilities to represent word similarity, which we haven’t explored:

- Modify the training objective for the pre-training phase, for example by adding a task which influences how the model represents word similarity.
- Fine-tune the model on a task to improve the representation of word similarity, for example predict the nearest neighbour based on existing similarity word lists.
- Replace wordpiece tokenization back to full word tokenization, which has been reported to improve performances in some contexts. [11]

On the other hand we didn’t spend much time to find the best parameters for the static embeddings and we just used a well established static embedding like fastText and didn’t test more recent proposals for static embeddings like [14] which reported improved results. So there is a lot of room for improvements in both directions.

In order to understand how the performance differences we observed between static and dynamic embeddings relate to the performance gains which have been observed by stacking embeddings from different sources [3], we combine word2vec, fastText and BERT embeddings in different constellations and add a fastText model with the same dimensions to compensate for effects based on the different dimensionality of the embeddings (see Figure 3). For four evaluation sets – GermaNet, Men, Duden, TOEFL – the differences between BERT and fastText are larger than the difference between fastText and a stacked alternative. The performance gain of using stacked embeddings is in most cases rather small. Adding BERT to the stacked embeddings either doesn’t help at all – TOEFL – or only a little bit – GermaNet, Schm280, Men, Duden. The only exception is the Wiktionary dataset which is already the only use case, where

Table 6: Performances of stacked embeddings in the seven discussed tasks. Top value per task is highlighted bold.

| | GermaNet (RC) | Wiktionary (RC) | SimLex999 (WS) | Schm280 (WR) | MEN (WR) | Duden (WC) | TOEFL (WC) |
|---------------------------|---------------|-----------------|----------------|-----------------|-----------------|-----------------|-------------|
| | dimensions | macro F1 | macro F1 | Spearman ρ | Spearman ρ | Spearman ρ | accuracy |
| BERT (sum-median-median) | 768 | .710 | .906 | .476 | .754 | .650 | .589 |
| FastText Dim768WS2 | 768 | .774 | .863 | .491 | .771 | .736 | .813 |
| FastText Dim1536WS2 | 1536 | .833 | .898 | .500 | .762 | .737 | .810 |
| Word2Vec Dim768 | | | | | | | |
| +FastText Dim768WS2 | 1536 | .826 | .859 | .479 | .765 | .733 | .820 |
| Word2Vec Dim768 | | | | | | | |
| +BERT (sum-median-median) | 1536 | .833 | .914 | .489 | .788 | .728 | .788 |
| FastText Dim768WS2 | | | | | | | |
| +BERT (sum-median-median) | 1536 | .791 | .930 | .508 | .802 | .734 | .637 |
| Word2Vec Dim768 | | | | | | | |
| +FastText Dim768WS2 | | | | | | | |
| +BERT (sum-median-median) | 2304 | .837 | .916 | .496 | .791 | .737 | .803 |

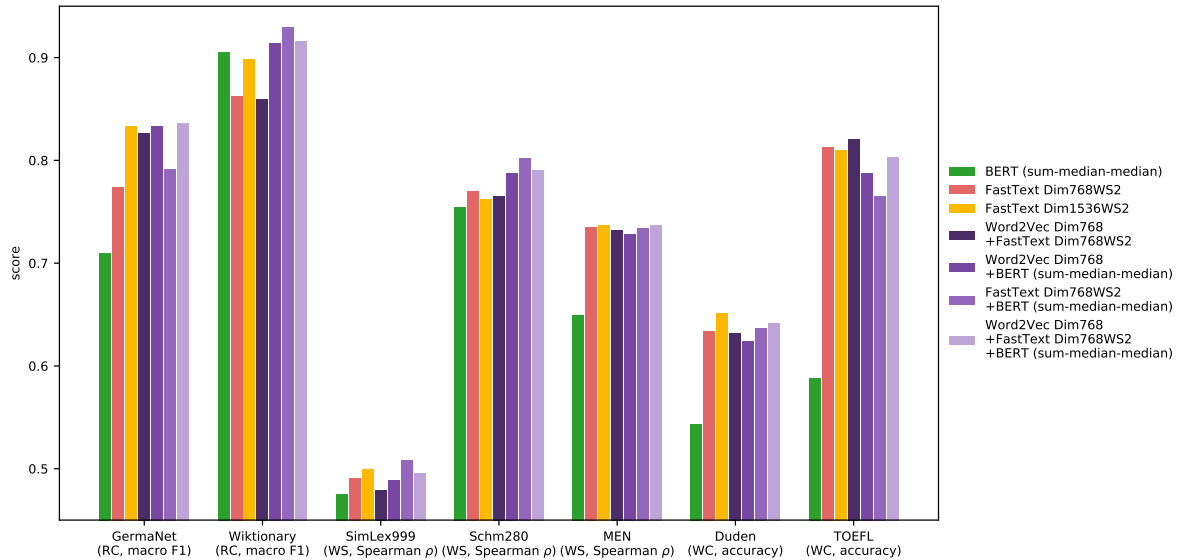


Figure 3: Performances of stacked embeddings in the seven discussed tasks. Graphical representation of the results in Table 6.

BERT is better than fastText. As discussed above the Wiktionary dataset consists mainly of inflections, for example singular vs. plural, or derivations, for example masculine form of a noun (*Autor*) vs. female form (*Autorin*). More examples are listed in Table 7. Maybe more sophisticated approaches combining the different embeddings like [13] will show better results, but obviously they all need a token-based model next to the static models.

Exploring the behaviour of the different embeddings we also came across a noticeable difference between the BERT-based embeddings and the static embeddings (see Figure 4). We calculated the distances between 956 synonym pairs, using synonyms as defined by GermaNet in one setting and defined by Duden in the other. To make the results comparable we standardized each of them by drawing 956 random word pairs and based our calculation of the mean distance and the standard deviation on them. Then we expressed the cosine distance of the synonyms in standard deviations away from the mean distance. The results show for both datasets a much larger spread for the static embeddings indicating that the BERT vectors occupy a smaller space, an effect which is not related to the dimensionality of its vectors.

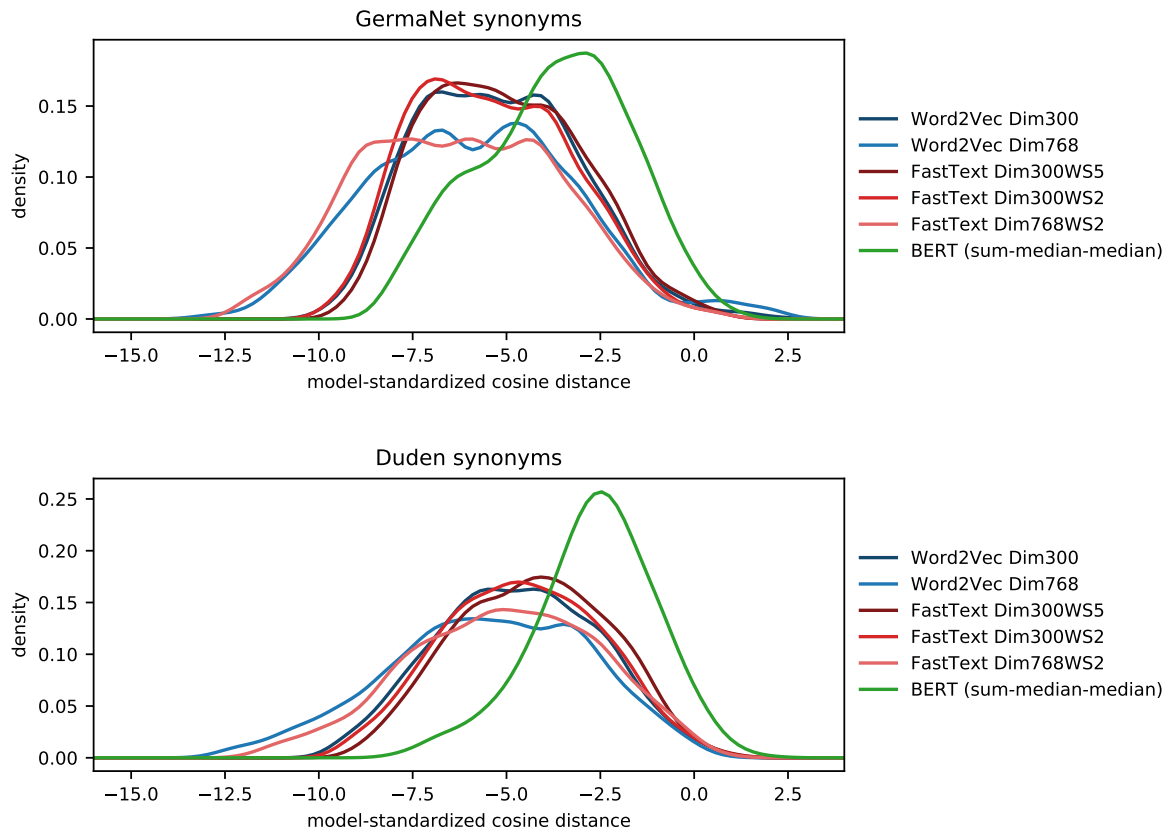


Figure 4: Kernel density estimation (Gaussian, $h = 0.5$) of the cosine distance of synonym pairs in the respective embeddings, each standardized for the respective embedding.

This seems to be in accordance with results from Ethayarajh [12], who reported that the contextualized token embedding of BERT is anisotropic: randomly sampled words seem to have, on average, a very high cosine similarity. In fact, Timkey and van Schijndel [37] report in a pre-print that in BERT’s contextualized embedding space, a few dimensions dominate similarity between word vectors (“rogue dimension”). As future work, we want to examine the effect of post-processing transformations on the embedding spaces, proposed by Timkey and van Schijndel, which are designed to counteract the undesirable effect of these rogue dimensions. In our first exploratory experiments we observe that all our examined embeddings – both the distilled ones from BERT, but also the static ones – appear to benefit from post-processing the type vectors. Yet even then, the post-processing still does not give BERT an advantage over static embeddings.

To summarize, our main take away is not a recommendation for a specific static word embedding, rather we think it is worthwhile to continue research on static word embeddings – at least for researchers working in the field of Computational Literary Studies –, because their representational power as abstractions of semantic systems is on par to that of dynamic embeddings, the needed computing power is much less and the minimal size of the corpora needed to train them is also smaller. What we need in the field of Computational Literary Studies is a more robust understanding how the quality of embeddings is related to the size and structure of datasets, methods to improve the performance of static embeddings trained on

even smaller datasets, maybe by combining them with knowledge bases, and more evaluation datasets for languages beyond English.

References

- [1] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. “On the Surprising Behavior of Distance Metrics in High Dimensional Space”. In: *Database Theory, ICDT 2001*. Ed. by J. Van den Bussche and V. Vianu. Lecture Notes in Computer Science. 2001, pp. 420–434. DOI: 10.1007/3-540-44503-x_27.
- [2] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa. “A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches”. In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Boulder, Colorado, 2009, pp. 19–27. URL: <https://aclanthology.org/N09-1003>.
- [3] A. Akbik, D. Blythe, and R. Vollgraf. “Contextual String Embeddings for Sequence Labeling”. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, 2018, pp. 1638–1649. URL: <https://aclanthology.org/C18-1139>.
- [4] A. Bakarov. “A survey of word embeddings evaluation methods”. In: *arXiv preprint arXiv:1801.09536* (2018). URL: <http://arxiv.org/abs/1801.09536>.
- [5] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon. “Time for a Change: a Tutorial for Comparing Multiple Classifiers Through Bayesian Analysis”. In: *Journal of Machine Learning Research* 18.77 (2017), pp. 1–36. URL: <http://jmlr.org/papers/v18/16-305.html>.
- [6] R. Bommasani, K. Davis, and C. Cardie. “Interpreting Pretrained Contextualized Representations via Reductions to Static Embeddings”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Online: Association for Computational Linguistics, 2020, pp. 4758–4781. DOI: 10.18653/v1/2020.acl-main.431.
- [7] E. Bruni, N.-K. Tran, and M. Baroni. “Multimodal distributional semantics”. In: *Journal of artificial intelligence research* 49 (2014), pp. 1–47. DOI: 10.1007/s10462-019-09796-3.
- [8] B. Chan, S. Schweter, and T. Möller. “German’s Next Language Model”. In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online), 2020, pp. 6788–6796. DOI: 10.18653/v1/2020.coling-main.598.
- [9] G. Corani, A. Benavoli, J. Demšar, F. Mangili, and M. Zaffalon. “Statistical comparison of classifiers through Bayesian hierarchical modelling”. In: *Machine Learning* 106.11 (2017), pp. 1817–1837. DOI: 10.1007/s10994-017-5641-9.
- [10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. NaacL-hlt 2019. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

- [11] H. El Boukkouri, O. Ferret, T. Lavergne, H. Noji, P. Zweigenbaum, and J. Tsujii. “CharacterBERT: Reconciling ELMo and BERT for Word-Level Open-Vocabulary Representations From Characters”. In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, 2020, pp. 6903–6915. DOI: 10.18653/v1/2020.coling-main.609.
- [12] K. Ethayarajh. “How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China, 2019, pp. 55–65. DOI: 10.18653/v1/D19-1006.
- [13] P. Gupta and M. Jaggi. “Obtaining Better Static Word Embeddings Using Contextual Embedding Models”. In: *arXiv preprint arXiv:2106.04302* (2021). URL: <http://arxiv.org/abs/2106.04302>.
- [14] P. Gupta, M. Pagliardini, and M. Jaggi. “Better Word Embeddings by Disentangling Contextual n-Gram Information”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota, 2019, pp. 933–939. DOI: 10.18653/v1/N19-1098.
- [15] B. Hamp and H. Feldweg. “GermaNet - a Lexical-Semantic Net for German”. In: *Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*. 1997. URL: <https://www.aclweb.org/anthology/W97-0802>.
- [16] S. Hengchen, R. Ros, and J. Marjanen. “A data-driven approach to the changing vocabulary of the ‘nation’ in English, Dutch, Swedish and Finnish newspapers, 1750-1950”. In: *Book of Abstracts of DH2019*. Utrecht, 2019. URL: <https://dev.clariah.nl/files/dh2019/boa/0791.html>.
- [17] V. Henrich and E. Hinrichs. “GernEdiT - The GermaNet Editing Tool”. In: *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC’10)*. Valletta, Malta: European Language Resources Association (ELRA), 2010, pp. 2228–2235. URL: <http://www.lrec-conf.org/proceedings/lrec2010/pdf/264%5C%5FPaper.pdf>.
- [18] F. Hill, R. Reichart, and A. Korhonen. “SimLex-999: Evaluating Semantic Models With (Genuine) Similarity Estimation”. In: *Computational Linguistics* 41.4 (2015), pp. 665–695. DOI: 10.1162/COLI_a_00237.
- [19] M. Kocher and J. Savoy. “Distributed language representation for authorship attribution”. In: *Digital Scholarship in the Humanities* 33.2 (2017), pp. 425–441. DOI: 10.1093/llc/fqx046.
- [20] M. Köper, C. Scheible, and S. Schulte im Walde. “Multilingual Reliability and “Semantic” Structure of Continuous Word Spaces”. In: *Proceedings of the 11th International Conference on Computational Semantics*. London, UK, 2015, pp. 40–45. URL: <https://aclanthology.org/W15-0105>.
- [21] V. Kulkarni, R. Al-Rfou, B. Perozzi, and S. Skiena. “Statistically Significant Detection of Linguistic Change”. In: *Proceedings of the 24th International World Wide Web Conference*. Www ’15. Florence, Italy, 2015, pp. 625–635. DOI: 10.1145/2736277.2741627.

- [22] A. Kutuzov, L. Øvrelid, T. Szymanski, and E. Velldal. “Diachronic word embeddings and semantic shifts: a survey”. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, 2018, pp. 1384–1397. URL: <https://aclanthology.org/C18-1117>.
- [23] A. Lenci, M. Sahlgren, P. Jeuniaux, A. C. Gyllensten, and M. Miliani. “A comprehensive comparative evaluation and analysis of Distributional Semantic Models”. In: *arXiv preprint arXiv:2105.09825* (2021). URL: <http://arxiv.org/abs/2105.09825>.
- [24] I. Leviant and R. Reichart. “Separated by an un-common language: Towards judgment language informed vector space modeling”. In: *arXiv preprint arXiv:1508.00106* (2015). URL: <http://arxiv.org/abs/1508.00106>.
- [25] O. Levy and Y. Goldberg. “Linguistic Regularities in Sparse and Explicit Word Representations”. In: *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. Proceedings of the Eighteenth Conference on Computational Natural Language Learning. Ann Arbor, Michigan: Association for Computational Linguistics, 2014, pp. 171–180. DOI: 10.3115/v1/W14-1618.
- [26] T. Mikolov, K. Chen, G. Corrado, and J. Dean. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013). URL: <http://arxiv.org/abs/1301.3781>.
- [27] T. Mikolov, W.-t. Yih, and G. Zweig. “Linguistic Regularities in Continuous Space Word Representations”. In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. NaacLHt 2013. Atlanta, Georgia: Association for Computational Linguistics, 2013, pp. 746–751. URL: <https://www.aclweb.org/anthology/N13-1090>.
- [28] P. J. Ortiz Suárez, L. Romary, and B. Sagot. “A Monolingual Approach to Contextualized Word Embeddings for Mid-Resource Languages”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020, pp. 1703–1714. URL: <https://www.aclweb.org/anthology/2020.acl-main.156>.
- [29] S. Rahmani, S. M. Fakhrahmad, and M. H. Sadreddini. “Co-occurrence graph-based context adaptation: a new unsupervised approach to word sense disambiguation”. In: *Digital Scholarship in the Humanities* (2020). DOI: 10.1093/llc/fqz048.
- [30] A. Rogers, O. Kovaleva, and A. Rumshisky. “A Primer in BERTology: What We Know About How BERT Works”. In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 842–866. DOI: 10.1162/tacl_a_00349.
- [31] R. Ros. “Conceptual Vocabularies and Changing Meanings of “Foreign” in Dutch Foreign News (1815–1914)”. In: *Book of Abstracts of DH2019*. Utrecht, 2019. URL: <https://dev.clariah.nl/files/dh2019/boa/0651.html>.
- [32] R. Ros and J. van Eijnatten. “Disentangling a Trinity: A Digital Approach to Modernity, Civilization and Europe in Dutch Newspapers (1840-1990)”. In: *Book of Abstracts of DH2019*. Utrecht, 2019. URL: <https://dev.clariah.nl/files/dh2019/boa/0572.html>.
- [33] D. Salami and S. Momtazi. “Recurrent convolutional neural networks for poet identification”. In: *Digital Scholarship in the Humanities* (2020). DOI: 10.1093/llc/fqz096.

- [34] Y. Song, T. Kimura, B. Batjargal, and A. Maeda. “Linking the Same Ukiyo-e Prints in Different Languages by Exploiting Word Semantic Relationships across Languages”. In: *Book of Abstracts of DH2017*. Alliance of Digital Humanities Organizations. Montréal, Canada, 2017. URL: <https://dh2017.adho.org/abstracts/369/369.pdf>.
- [35] Y. Susanti, T. Tokunaga, H. Nishikawa, and H. Obari. “Automatic distractor generation for multiple-choice English vocabulary questions”. In: *Research and Practice in Technology Enhanced Learning* 13.2 (2018). DOI: 10.1186/s41039-018-0082-z.
- [36] M. A. H. Taieb, T. Zesch, and M. B. Aouicha. “A survey of semantic relatedness evaluation datasets and procedures”. In: *Artificial Intelligence Review* 53.6 (2020), pp. 4407–4448.
- [37] W. Timkey and M. van Schijndel. “All Bark and No Bite: Rogue Dimensions in Transformer Language Models Obscure Representational Quality”. In: *arXiv:2109.04404 [cs]* (2021). URL: <http://arxiv.org/abs/2109.04404>.
- [38] T. Uslu, A. Mehler, C. Schulz, and D. Baumartz. “BigSense: a Word Sense Disambiguator for Big Data”. In: *Book of Abstracts of DH2019*. Utrecht, 2019. URL: <https://dev.clariah.nl/files/dh2019/boa/0199.html>.
- [39] I. Vulić, E. M. Ponti, R. Litschko, G. Glavaš, and A. Korhonen. “Probing Pretrained Language Models for Lexical Semantics”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Online: Association for Computational Linguistics, 2020, pp. 7222–7240. DOI: 10.18653/v1/2020.emnlp-main.586.
- [40] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding”. In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 353–355. DOI: 10.18653/v1/W18-5446.
- [41] B. Wang, A. Wang, F. Chen, Y. Wang, and C.-C. J. Kuo. “Evaluating word embedding models: methods and experimental results”. In: *APSIPA transactions on signal and information processing* 8 (2019). DOI: 10.1017/atsip.2019.12.
- [42] Y. Wang, L. Cui, and Y. Zhang. “How Can BERT Help Lexical Semantics Tasks?” In: *arXiv preprint arXiv:1911.02929* (2020). URL: <http://arxiv.org/abs/1911.02929>.
- [43] S. Ziehe and C. Sporleder. “Multimodale Sentimentanalyse politischer Tweets”. In: *Book of Abstracts of DHd2019*. Frankfurt, 2019, pp. 331–332. DOI: 10.5281/zenodo.2596095.

A. Appendix

A.1. Supplementary tables and figures

Table 7

Individual relations of the GermaNet and Wiktionary dataset.

| relation | # instances | examples |
|---|-------------|--|
| germanet_lexrel_has_pertainym | 1290 | (erfolgreich, Erfolg), (problemlos, Problem), (unterschiedlich, Unterschied) |
| germanet_lexrel_is_part_of | 193 | (Hotelzimmer, Hotel), (Handgelenk, Hand), (Autoradio, Auto) |
| germanet_lexrel_has_material | 176 | (Teppichboden, Teppich), (Lederjacke, Leder), (Acrylglas, Acryl) |
| germanet_lexrel_has_part | 165 | (Treppenhaus, Treppe), (Zifferblatt, Ziffer), (Hafenstadt, Hafen) |
| germanet_lexrel_has_user | 150 | (Tierarzt, Tier), (Tierheim, Tier), (Krankenhaus, Kranke) |
| germanet_lexrel_has_location | 129 | (Gartenhaus, Garten), (Fußbodenheizung, Fußboden), (Dachboden, Dach) |
| germanet_lexrel_has_active_usage | 119 | (Rettungswagen, Rettung), (Kreuzfahrtschiff, Kreuzfahrt), (Schlafanzug, Schlaf) |
| germanet_lexrel_has_specialization | 107 | (Projektleiter, Projekt), (Reiseleiter, Reise), (Jugendleiter, Jugendgruppe) |
| germanet_lexrel_has_usage | 107 | (Autobahn, Auto), (Gotteshaus, Gottesdienst), (Kurhaus, Kur) |
| germanet_lexrel_has_purpose_of_usage | 100 | (Couchtisch, Couch), (Handtuch, Hand), (Kotflügel, Kot) |
| germanet_lexrel_has_topic | 99 | (Reisebüro, Reise), (Berufsschule, Beruf), (Liebesroman, Liebe) |
| germanet_lexrel_is_container_for | 84 | (Briefkasten, Brief), (Mülleimer, Müll), (Mülltonne, Müll) |
| germanet_lexrel_has_raw_product | 83 | (Zitronensaft, Zitrone), (Kokosöl, Kokosnuss), (Kokosmilch, Kokosnuss) |
| germanet_lexrel_has_manner_of_functioning | 78 | (Seilbahn, Seil), (Atombombe, Atom), (Handbremse, Hand) |
| germanet_lexrel_has_appearance | 76 | (Wendeltreppe, Wendel), (Ringelblume, Ringel), (Plüschtier, Tier) |
| germanet_lexrel_has_attribute | 76 | (Ferienwohnung, Ferien), (Ferienhaus, Ferien), (Handbuch, Hand) |
| germanet_lexrel_has_ingredient | 76 | (Currywurst, Curry), (Hefeteig, Hefe), (Vollkornbrot, Vollkorn) |
| germanet_lexrel_has_function | 73 | (Chefarzt, Chef), (Chefkoch, Chef), (Liegestuhl, Liege) |
| germanet_lexrel_has_time | 73 | (Wintergarten, Winter), (Winterreifen, Winter), (Nachtisch, Nacht) |
| germanet_lexrel_has_product | 62 | (Honigbiene, Honig), (Textilfabrik, Textil), (Obstbaum, Obst) |
| germanet_lexrel_has_origin | 61 | (Regenwasser, Regen), (Leserbrief, Leser), (Menschensohn, Mensch) |
| germanet_lexrel_has_content | 57 | (Telefonbuch, Telefonnummer), (Branchenbuch, Branche), (Terminkalender, Termin) |
| germanet_lexrel_has_owner | 56 | (Bundesstraße, Bundesrepublik), (Vereinsheim, Verein), (Feuerwehrhaus, Feuerwehr) |
| germanet_lexrel_has_habitat | 53 | (Tiergarten, Tier), (Zimmerpflanze, Zimmer), (Alpenrose, Alpen) |
| germanet_lexrel_has_prototypical_place_of_usage | 39 | (Gartenmöbel, Garten), (Raumschiff, Weltraum), (Geländewagen, Gelände) |
| germanet_lexrel_has_occasion | 34 | (Ehering, Ehe), (Schultüte, Schulbeginn), (Neujahrskonzert, Neujahr) |
| germanet_lexrel_has_member | 32 | (Ingenieurbüro, Ingenieur), (Abgeordnetenhaus, Abgeordnete), (Hotelkette, Hotel) |
| germanet_lexrel_is_location_of | 30 | (Firmengelände, Firma), (Vereinsgelände, Verein), (Museumsinsel, Museum) |
| germanet_lexrel_has_other_property | 30 | (Blutzucker, Blut), (Blutbad, Blut), (Hosenanzug, Hose) |
| germanet_lexrel_has_component | 29 | (Schmutzwasser, Schmutz), (Seifenwasser, Seife), (Duftöl, Duftstoff) |
| germanet_lexrel_has_goods | 29 | (Autohaus, Auto), (Biergarten, Bier), (Warenhaus, Ware) |
| germanet_lexrel_is_storage_for | 28 | (Bootshaus, Boot), (Gemäldegalerie, Gemälde), (Gewandhaus, Gewand) |
| germanet_lexrel_has_prototypical_holder | 28 | (Handschuh, Hand), (Ohrstecker, Ohr), (Ohrring, Ohr) |
| germanet_lexrel_is_member_of | 28 | (Ehefrau, Ehe), (Ehemann, Ehe), (Hansstadt, Hanse) |
| germanet_lexrel_has_eponym | 24 | (Marienkirche, Maria), (Disneyland, Disney), (Marienkäfer, Maria) |
| germanet_lexrel_has_relation | 21 | (Kreisstadt, Kreisverwaltung), (Gruppenleiter, Gruppe), (Torschützenkönig, Torschütze) |
| germanet_lexrel_has_production_method | 19 | (Baggersee, Bagger), (Recyclingpapier, Recycling), (Sägemehl, Säge) |
| germanet_lexrel_is_product_of | 19 | (Spinnennetz, Spinne), (Wespennest, Wespe), (Storchennest, Storch) |
| germanet_lexrel_has_consistency_of | 13 | (Puderzucker, Puder), (Honigmelone, Honig), (Krepppapier, Krepp) |
| germanet_lexrel_is_comparable_to | 12 | (Mammutbaum, Mammut), (Hitzkopf, Hitze), (Inselberg, Insel) |
| germanet_lexrel_has_no_property | 11 | (Zaunkönig, Zaun), (Cocktailtomate, Cocktail), (Lackaffe, Lack) |
| germanet_lexrel_is_prototypical_holder_for | 11 | (Glockenturm, Glocke), (Gardinenstange, Gardine), (Fahrradbügel, Fahrrad) |
| wiktionary_derivations_subst_in | 2000 | (Autor, Autorin), (Sänger, Sängerin), (Schauspieler, Schauspielerin) |
| wiktionary_derivations_subst_ung | 1059 | (nutzen, Nutzung), (unterstützen, Unterstützung), (meinen, Meinung) |
| wiktionary_derivations_adj_isch | 566 | (Telefon, telefonisch), (England, englisch), (Medizin, medizinisch) |
| wiktionary_derivations_subst_er | 507 | (nutzen, Nutzer), (besuchen, Besucher), (lesen, Leser) |
| wiktionary_derivations_subst_keit | 333 | (tätig, Tätigkeit), (wirklich, Wirklichkeit), (verfügbar, Verfügbarkeit) |
| wiktionary_derivations_adj_lich | 235 | (Natur, natürlich), (Ende, endlich), (Person, persönlich) |
| wiktionary_derivations_adj_ig | 202 | (Stand, ständig), (Zustand, zuständig), (Kraft, kräftig) |
| wiktionary_derivations_adj_los | 192 | (Kosten, kostenlos), (Problem, problemlos), (Mühe, mühelos) |
| wiktionary_derivations_subst_chen | 186 | (Brot, Brötchen), (Paar, Pärchen), (Mann, Männchen) |
| wiktionary_derivations_subst_heit | 111 | (sicher, Sicherheit), (mehr, Mehrheit), (gelegen, Gelegenheit) |
| wiktionary_derivations_adj_bar | 87 | (verfügen, verfügbar), (erkennen, erkennbar), (denken, denkbar) |
| wiktionary_derivations_subst_e | 82 | (helfen, Hilfe), (erst, Erste), (anzeigen, Anzeige) |
| wiktionary_derivations_subst_ei | 67 | (Bäcker, Bäckerei), (Gärtner, Gärtnerei), (Brenner, Brennerei) |
| wiktionary_derivations_subst_schaft | 51 | (Partner, Partnerschaft), (Mitglied, Mitgliedschaft), (Meister, Meisterschaft) |
| wiktionary_derivations_adj_haft | 45 | (Beispiel, beispielhaft), (Masse, massenhaft), (Zweifel, zweifelhaft) |
| wiktionary_derivations_subst_lein | 40 | (Buch, Büchlein), (Bauch, Bäuchlein), (Licht, Lichtlein) |
| wiktionary_derivations_subst_ler | 38 | (Wissenschaft, Wissenschaftler), (Sport, Sportler) |
| wiktionary_derivations_subst_tum | 28 | (Christ, Christentum), (Jude, Judentum), (Brauch, Brauchtum) |
| wiktionary_derivations_subst_ling | 28 | (früh, Frühling), (neu, Neuling), (flüchten, Flüchtling) |
| wiktionary_derivations_adj_en | 19 | (Perle, perlen), (Metall, metallen), (Bronze, bronzen) |
| wiktionary_derivations_subst_nis | 19 | (erleben, Erlebnis), (verhalten, Verhältnis), (erlauben, Erlaubnis) |
| wiktionary_derivations_adj_fach | 18 | (drei, dreifach), (zwei, zweifach), (vier, vierfach) |
| wiktionary_derivations_adj_sam | 17 | (wirken, wirksam), (Rat, ratsam), (gemein, gemeinsam) |
| wiktionary_infl_verb_partizip_perfekt | 2115 | (machen, gemacht), (finden, gefunden), (sein, gewesen) |
| wiktionary_infl_verb_sg_2p_präsens | 2014 | (können, kannst), (haben, hast), (sein, bist) |
| wiktionary_infl_verb_sg_1p_präsens | 2003 | (einen, eine), (haben, habe), (sein, bin) |
| wiktionary_infl_adj_komparativ | 2002 | (viel, mehr), (wenig, weniger), (weit, weiter) |
| wiktionary_infl_subst_nom_pl | 2001 | (Jahr, Jahre), (Bild, Bilder), (Kind, Kinder) |
| wiktionary_infl_subst_gen_sg | 2001 | (Foto, Fotos), (Jahr, Jahres), (Video, Videos) |
| wiktionary_infl_subst_dat_pl | 1998 | (Jahr, Jahren), (Kind, Kindern), (Spiel, Spielen) |
| wiktionary_infl_adj_superlativ | 1997 | (viel, meisten), (wichtig, wichtigsten), (groß, größten) |
| wiktionary_infl_verb_sg_1p_prät_indikativ | 643 | (sein, war), (werden, wurde), (haben, hatte) |
| wiktionary_infl_verb_sg_1p_prät_konjunktiv | 470 | (werden, würde), (sein, wäre), (können, könnte) |

Table 8

Comparison of the embeddings computed by the different distillation methods on a subset of the MEN task (WR), TOEFL task (WC), Wiktionary task, and Germanet task (RC). Reported numbers are the mean standardized score over all four tasks. Row maxima are highlighted bold. The top 13 maxima are underlined.

| poolings | vectorization aggregations | inputemb | L1 | L2 | L3 | L4 | L5 | L6 | L7 | L8 | L9 | L10 | L11 | L12 | L1-4 | L9-12 | sum | all |
|------------|----------------------------|----------|--------|--------|--------------|--------------|--------|--------|--------|--------|--------|--------|--------|--------|--------------|--------|--------|--------------|
| first | l0.5medoid | -3.453 | -1.850 | -0.831 | -0.090 | 0.102 | -0.524 | -0.583 | -0.839 | -0.743 | -0.966 | -0.595 | -0.403 | -0.261 | -0.119 | -0.206 | -0.018 | 0.474 |
| | l1medoid | -3.354 | -1.926 | -0.838 | -0.052 | 0.177 | -0.447 | -0.626 | -0.896 | -0.673 | -0.974 | -0.563 | -0.494 | -0.256 | -0.184 | -0.184 | 0.019 | 0.437 |
| | l2medoid | -3.422 | -1.812 | -0.909 | -0.099 | 0.223 | -0.342 | -0.597 | -0.851 | -0.746 | -0.879 | -0.649 | -0.392 | -0.169 | -0.224 | -0.202 | -0.045 | 0.403 |
| | mean | -2.698 | -1.005 | -0.831 | 0.085 | 0.221 | -0.198 | -0.293 | -0.670 | -0.445 | -0.784 | -0.523 | -0.247 | -0.450 | -0.021 | -0.255 | -0.146 | 0.072 |
| | meannorm | -2.719 | -0.998 | -0.821 | 0.015 | 0.166 | -0.211 | -0.316 | -0.709 | -0.480 | -0.815 | -0.521 | -0.271 | -0.382 | -0.022 | -0.285 | -0.210 | 0.031 |
| | median | -2.459 | -0.969 | -0.817 | 0.143 | 0.212 | -0.216 | -0.291 | -0.621 | -0.448 | -0.752 | -0.492 | -0.291 | -0.478 | 0.016 | -0.257 | -0.089 | 0.116 |
| last | l0.5medoid | -0.979 | -0.492 | -0.411 | 0.009 | 0.188 | -0.002 | -0.247 | -0.399 | -0.585 | -0.394 | -0.263 | -0.285 | -0.401 | 0.046 | -0.021 | 0.206 | 0.369 |
| | l1medoid | -1.007 | -0.608 | -0.398 | 0.063 | 0.249 | 0.060 | -0.238 | -0.413 | -0.471 | -0.340 | -0.216 | -0.299 | -0.404 | 0.106 | -0.009 | 0.171 | 0.327 |
| | l2medoid | -0.922 | -0.638 | -0.396 | 0.043 | 0.327 | 0.016 | -0.177 | -0.453 | -0.444 | -0.429 | -0.194 | -0.259 | -0.266 | 0.179 | -0.126 | 0.079 | 0.389 |
| | mean | -0.689 | -0.544 | -0.436 | 0.162 | 0.232 | 0.153 | -0.042 | -0.043 | -0.158 | -0.101 | -0.103 | -0.162 | -0.381 | -0.002 | -0.057 | 0.229 | 0.344 |
| | meannorm | -0.858 | -0.589 | -0.405 | 0.188 | 0.302 | 0.185 | -0.005 | -0.012 | -0.123 | -0.061 | -0.008 | -0.077 | -0.333 | 0.002 | -0.009 | 0.251 | 0.392 |
| | median | -0.674 | -0.570 | -0.420 | 0.160 | 0.271 | 0.146 | -0.019 | -0.040 | -0.178 | -0.076 | -0.057 | -0.152 | -0.363 | 0.031 | -0.049 | 0.279 | 0.392 |
| l0.5medoid | l0.5medoid | -2.517 | -1.359 | -0.331 | 0.223 | 0.142 | -0.190 | -0.321 | -0.309 | -0.488 | -0.590 | -0.382 | -0.304 | -0.251 | 0.159 | -0.162 | 0.263 | 0.629 |
| | l1medoid | -1.007 | -1.397 | -0.345 | 0.193 | 0.144 | -0.040 | -0.261 | -0.410 | -0.493 | -0.663 | -0.392 | -0.374 | -0.260 | 0.147 | -0.133 | 0.344 | 0.473 |
| | l2medoid | -2.505 | -1.310 | -0.381 | 0.234 | 0.215 | -0.104 | -0.254 | -0.336 | -0.536 | -0.726 | -0.424 | -0.272 | -0.182 | 0.023 | -0.198 | 0.160 | 0.426 |
| | mean | -2.052 | -0.825 | -0.240 | 0.288 | 0.367 | 0.130 | 0.007 | -0.316 | -0.214 | -0.388 | -0.283 | 0.014 | -0.313 | 0.087 | -0.174 | 0.147 | 0.189 |
| | meannorm | -2.071 | -0.875 | -0.304 | 0.320 | 0.323 | 0.120 | -0.021 | -0.407 | -0.237 | -0.372 | -0.247 | 0.025 | -0.240 | 0.090 | -0.162 | 0.157 | 0.223 |
| | median | -1.926 | -0.832 | -0.316 | 0.258 | 0.287 | 0.080 | -0.075 | -0.340 | -0.334 | -0.361 | -0.225 | -0.054 | -0.373 | 0.099 | -0.215 | 0.168 | 0.222 |
| l1medoid | l0.5medoid | -2.566 | -1.344 | -0.416 | 0.165 | 0.115 | -0.135 | -0.346 | -0.271 | -0.481 | -0.604 | -0.388 | -0.288 | -0.265 | 0.123 | -0.105 | 0.267 | 0.606 |
| | l1medoid | -2.546 | -1.461 | -0.528 | 0.165 | 0.134 | -0.080 | -0.300 | -0.393 | -0.497 | -0.668 | -0.425 | -0.358 | -0.232 | 0.125 | -0.110 | 0.324 | 0.456 |
| | l2medoid | -2.539 | -1.301 | -0.453 | 0.189 | 0.215 | -0.076 | -0.283 | -0.331 | -0.507 | -0.712 | -0.424 | -0.264 | -0.198 | 0.051 | -0.207 | 0.131 | 0.438 |
| | mean | -2.146 | -0.938 | -0.240 | 0.212 | 0.354 | 0.077 | -0.006 | -0.278 | -0.229 | -0.399 | -0.333 | 0.004 | -0.391 | 0.080 | -0.188 | 0.133 | 0.206 |
| | meannorm | -2.117 | -0.969 | -0.294 | 0.238 | 0.323 | 0.060 | -0.015 | -0.337 | -0.202 | -0.403 | -0.323 | -0.008 | -0.267 | 0.030 | -0.139 | 0.130 | 0.250 |
| | median | -1.983 | -0.949 | -0.289 | 0.198 | 0.256 | 0.093 | -0.012 | -0.358 | -0.256 | -0.390 | -0.206 | -0.113 | -0.415 | 0.086 | -0.201 | 0.084 | 0.212 |
| l2medoid | l0.5medoid | -2.512 | -1.438 | -0.355 | 0.115 | 0.233 | -0.240 | -0.329 | -0.339 | -0.489 | -0.670 | -0.362 | -0.215 | -0.184 | 0.219 | -0.153 | 0.341 | 0.653 |
| | l1medoid | -2.545 | -1.502 | -0.430 | 0.175 | 0.229 | -0.106 | -0.305 | -0.419 | -0.515 | -0.742 | -0.416 | -0.320 | -0.240 | 0.160 | -0.094 | 0.345 | 0.549 |
| | l2medoid | -2.555 | -1.459 | -0.411 | 0.170 | 0.328 | -0.162 | -0.238 | -0.380 | -0.521 | -0.729 | -0.444 | -0.217 | -0.218 | 0.096 | -0.174 | 0.251 | 0.564 |
| | mean | -2.240 | -0.973 | -0.299 | 0.286 | 0.302 | 0.061 | -0.045 | -0.265 | -0.277 | -0.455 | -0.250 | -0.065 | -0.338 | 0.104 | -0.200 | 0.164 | 0.226 |
| | meannorm | -2.293 | -0.980 | -0.332 | 0.288 | 0.284 | 0.087 | -0.097 | -0.328 | -0.268 | -0.457 | -0.236 | -0.021 | -0.240 | 0.090 | -0.139 | 0.174 | 0.249 |
| | median | -2.021 | -1.082 | -0.361 | 0.203 | 0.237 | 0.054 | -0.118 | -0.300 | -0.286 | -0.413 | -0.190 | -0.119 | -0.389 | 0.067 | -0.177 | 0.124 | 0.279 |
| nopooling | l0.5medoid | -1.936 | -1.397 | -0.909 | -0.024 | 0.019 | -0.260 | -0.621 | -0.901 | -0.594 | -0.674 | -0.315 | -0.037 | -0.251 | -0.259 | -0.093 | 0.022 | 0.284 |
| | l1medoid | -1.804 | -1.379 | -1.004 | -0.011 | -0.050 | -0.113 | -0.579 | -0.978 | -0.587 | -0.595 | -0.281 | -0.155 | -0.298 | -0.227 | -0.210 | 0.050 | 0.295 |
| | l2medoid | -1.587 | -1.437 | -0.920 | -0.035 | 0.093 | -0.260 | -0.514 | -0.844 | -0.696 | -0.687 | -0.342 | -0.156 | -0.203 | -0.212 | -0.144 | 0.042 | 0.171 |
| | mean | 0.567 | 0.682 | 0.752 | 1.185 | 1.077 | 0.900 | 0.636 | 0.483 | 0.313 | 0.309 | 0.477 | 0.661 | 0.448 | 1.122 | 0.789 | 0.927 | 1.095 |
| | meannorm | 0.061 | 0.398 | 0.526 | 1.115 | 1.055 | 0.869 | 0.607 | 0.469 | 0.305 | 0.323 | 0.459 | 0.701 | 0.486 | 0.964 | 0.751 | 0.898 | 1.053 |
| | median | 0.519 | 0.621 | 0.717 | 1.157 | 1.128 | 0.897 | 0.666 | 0.422 | 0.344 | 0.247 | 0.466 | 0.670 | 0.511 | 1.206 | 0.795 | 1.018 | 1.122 |
| mean | l0.5medoid | -0.404 | 0.291 | 0.292 | 0.761 | 0.805 | 0.338 | 0.070 | -0.206 | -0.376 | -0.272 | 0.014 | 0.281 | 0.127 | 0.973 | 0.325 | 0.658 | 0.797 |
| | l1medoid | -0.483 | 0.214 | 0.297 | 0.820 | 0.823 | 0.391 | -0.003 | -0.260 | -0.328 | -0.286 | 0.059 | 0.253 | 0.158 | 0.907 | 0.250 | 0.675 | 0.740 |
| | l2medoid | -0.361 | 0.309 | 0.353 | 0.737 | 0.870 | 0.331 | 0.119 | -0.347 | -0.296 | -0.166 | -0.026 | 0.215 | 0.202 | 0.964 | 0.233 | 0.557 | 0.769 |
| | mean | 0.567 | 0.682 | 0.752 | 1.185 | 1.077 | 0.900 | 0.636 | 0.483 | 0.313 | 0.309 | 0.477 | 0.661 | 0.448 | 1.122 | 0.789 | 0.927 | 1.095 |
| | meannorm | 0.188 | 0.436 | 0.575 | 1.131 | 1.067 | 0.851 | 0.633 | 0.468 | 0.318 | 0.321 | 0.459 | 0.662 | 0.502 | 0.970 | 0.766 | 0.887 | 1.042 |
| | median | 0.590 | 0.683 | 0.704 | 1.179 | 1.099 | 0.834 | 0.613 | 0.501 | 0.309 | 0.300 | 0.404 | 0.691 | 0.539 | 1.101 | 0.795 | 0.959 | 1.100 |
| meannorm | l0.5medoid | -0.632 | 0.045 | 0.184 | 0.646 | 0.857 | 0.351 | 0.272 | -0.249 | -0.252 | -0.184 | -0.040 | 0.208 | 0.201 | 0.817 | 0.305 | 0.701 | 0.751 |
| | l1medoid | -0.685 | 0.002 | 0.224 | 0.693 | 0.926 | 0.399 | 0.126 | -0.092 | -0.232 | -0.175 | -0.034 | 0.273 | 0.198 | 0.821 | 0.348 | 0.697 | 0.730 |
| | l2medoid | -0.621 | 0.032 | 0.321 | 0.673 | 0.904 | 0.560 | 0.077 | -0.140 | -0.216 | -0.180 | 0.000 | 0.444 | 0.227 | 0.768 | 0.487 | 0.591 | 0.655 |
| | mean | 0.122 | 0.403 | 0.519 | 1.107 | 1.049 | 0.827 | 0.603 | 0.441 | 0.317 | 0.321 | 0.455 | 0.635 | 0.466 | 0.950 | 0.654 | 0.864 | 1.018 |
| | meannorm | 0.066 | 0.397 | 0.527 | 1.113 | 1.054 | 0.868 | 0.603 | 0.475 | 0.306 | 0.332 | 0.458 | 0.685 | 0.516 | 0.966 | 0.750 | 0.884 | 1.053 |
| | median | 0.171 | 0.411 | 0.493 | 1.080 | 1.012 | 0.811 | 0.644 | 0.410 | 0.344 | 0.263 | 0.409 | 0.676 | 0.511 | 0.939 | 0.697 | 0.872 | 1.021 |
| median | l0.5medoid | -0.501 | 0.368 | 0.305 | 0.833 | 0.745 | 0.457 | 0.139 | -0.129 | -0.193 | -0.286 | -0.021 | 0.342 | 0.229 | 0.940 | 0.358 | 0.936 | 0.998 |
| | l1medoid | -0.442 | 0.241 | 0.306 | 0.927 | 0.775 | 0.558 | 0.103 | -0.189 | -0.147 | -0.261 | 0.065 | 0.310 | 0.193 | 0.997 | 0.377 | 0.884 | 0.886 |
| | l2medoid | -0.344 | 0.366 | 0.354 | 0.900 | 0.830 | 0.579 | 0.274 | -0.270 | -0.153 | -0.207 | 0.094 | 0.326 | 0.203 | 1.085 | 0.393 | 0.675 | 0.916 |
| | mean | 0.511 | 0.716 | 0.763 | 1.121 | 1.139 | 1.007 | 0.711 | 0.592 | 0.262 | 0.316 | 0.561 | 0.681 | 0.455 | 1.251 | 0.747 | 1.074 | 1.111 |
| | meannorm | 0.184 | 0.507 | 0.636 | 1.101 | 1.124 | 0.951 | 0.702 | 0.535 | 0.298 | 0.306 | 0.557 | 0.649 | 0.496 | 1.129 | 0.759 | 1.074 | 1.110 |
| | median | 0.549 | 0.694 | 0.739 | 1.147 | 1.106 | 0.975 | 0.708 | 0.536 | 0.368 | 0.287 | 0.529 | 0.729 | 0.529 | 1.246 | 0.778 | 1.082 | 1.169 |

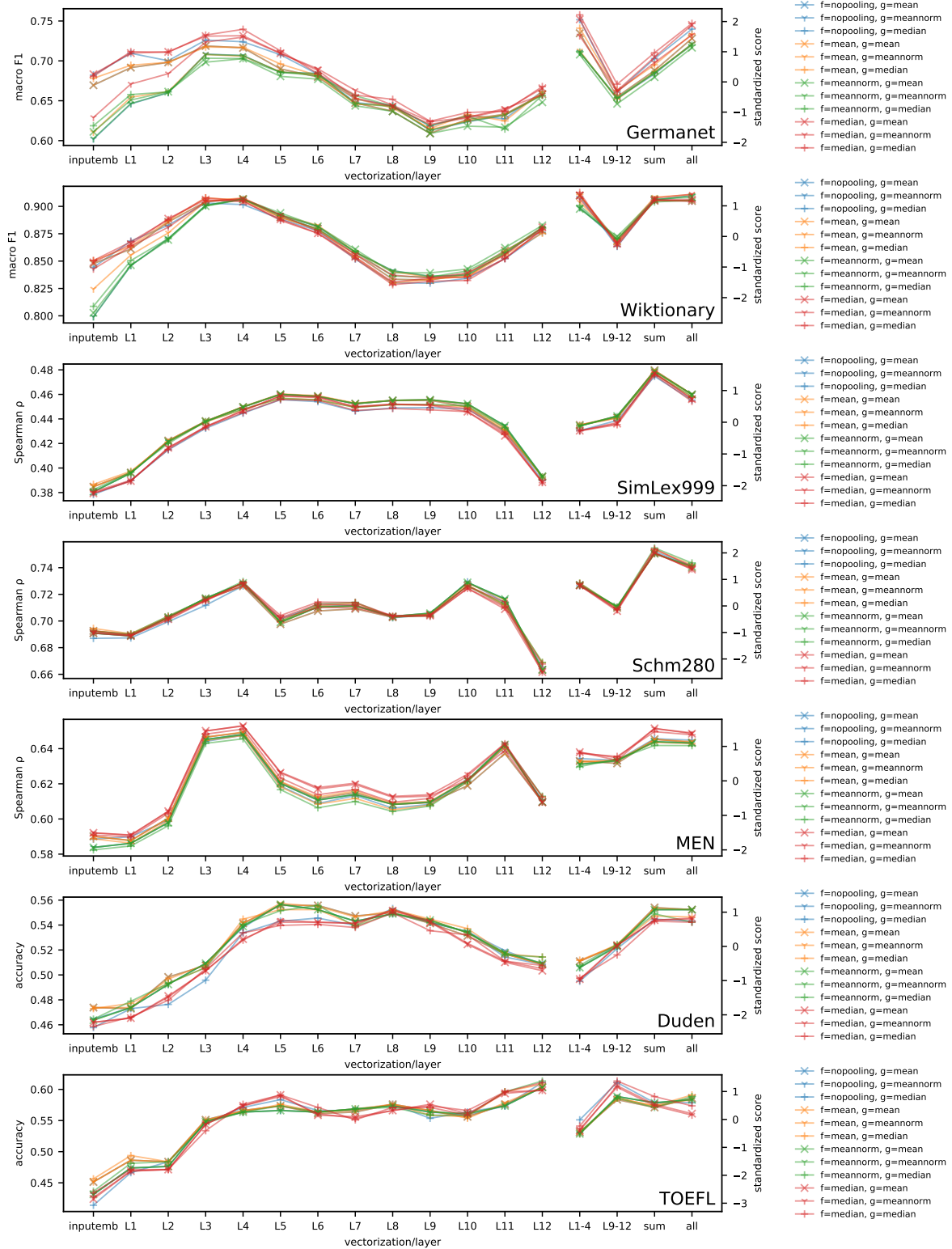


Figure 5: Comparison of the embeddings computed by the medoid-based distillation methods on the full evaluation dataset, visualized for each dataset separately.

A.2. Bayesian model comparison

To compare two embeddings on our datasets, we have employed the Bayesian hierarchical correlated t-test as described by Corani et al. [9]. This test was originally designed to compare two classifiers on multiple datasets, given their respective *cross-validation* results.

As presented in Sec. 3.3, our tasks do not perform such cross-validation. Therefore, to adapt to the test, we modify our tasks as follows to obtain cross-validation results:

- As the Relation Classification task (**Germanet**, **Wiktionary**) is implemented as median-based 1-nearest-neighbor classifier, it can be naturally extended to separate train and test sets. Given a train set of (labeled) word pairs and a test set of word pairs, we construct the decision objects (i.e., median) for the relations only on the training examples. Then we test the 1-nearest-neighbor classifier only on the test examples.

On each Relation Classification dataset, we perform 10 runs of 10-fold stratified cross validations to obtain 100 F1 scores.

- For the Word Relatedness and Word Similarity tasks (**SimLex999**, **Schm280**, **MEN**), there is no natural way to implement a cross-validation, since these tasks measure correlation and are not “trained”.

Therefore, to mimic the 10-fold cross-validation, on each dataset we randomly sample 100 subsets that each contain 10% of the respective dataset, and calculate the Spearman ρ on each of these subsets to obtain 100 correlation coefficients.

- Similarly for the Word Choice tasks (**Duden**, **TOEFL**). We randomly sample 100 subsets containing 10% of the respective dataset, and calculate the accuracies on each subset.

Fix a pair of two models we want to compare. For i -th dataset (of a total of q datasets), we calculate a vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{i100})$ of *differences of score* on each cross-validation fold, using the same fold for each dataset. On these vectors $\mathbf{x}_1, \dots, \mathbf{x}_q$, we now can perform the Bayesian hierarchical correlated t-test using the Python package `baycomp`⁷ that implements the hierarchical stochastic model and performs the “hypothesis tests” that estimates the posterior distribution of the difference of score between the two models on a future unseen data set, as proposed by Corani et al.

⁷<https://github.com/janezd/baycomp>